НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«**До захисту допущено**»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *Комплекс задач агрегації та аналізу новин*

Виконав: студент 4 курсу, групи ІП-52

Молявчик Владислав Михайлович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викладач Ковтунець О.В.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц каф. ТК, к.т.н. Ткаченко В.В.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Молявчику Владиславу Михайловичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Комплекс задач агрегації та аналізу новин»*

керівник проекту *Ковтунець Олесь Володимирович, ст. викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” квітня 2019 р. №1181-с

2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: опис та аналіз предметної області, аналіз існуючих проектів, розроблення функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки даних
3) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис методик та процесів тестування, опис контрольного прикладу
4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, робота з програмним забезпеченням

5. Перелік графічного матеріалу

1) Схема бази даних

2) Схема структурна класів програмного забезпечення

3) Схема структурна варіантів використань

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «12» *березня 2018 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*18.03.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*25.03.2019*	
3.	*Постановка та формалізація задачі*	*25.03.2019*	
4.	*Аналіз вимог до програмного забезпечення*	*01.04.2019*	
5.	*Алгоритмізація задачі*	*01.04.2019*	
6.	*Моделювання програмного забезпечення*	*08.04.2019*	
7.	*Обґрунтування використовуваних технічних засобів*	*15.04.2019*	
8.	*Розробка архітектури програмного забезпечення*	*22.04.2019*	
9.	*Розробка програмного забезпечення*	*29.04.2019*	
10.	*Налагодження програми*	*06.05.2019*	
11.	*Виконання графічних документів*	*13.05.2019*	
12.	*Оформлення пояснювальної записки*	*20.05.2019*	
13.	*Подання ДП на попередній захист*	*28.05.2019*	
14.	*Подання ДП рецензенту*	*03.05.2019*	
15.	*Подання ДП на основний захист*	*08.06.2019*	

Студент _____ Молявчик В.М.
(підпис)

Керівник проекту _____ Ковтунець О.В.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проект	2	
2	A4	КПІ.ІП-5211.045440.02.81	Пояснювальна записка	73	
3	A4	КПІ.ІП-5211.045440.03.91	Технічне завдання	11	
4	A4	КПІ.ІП-5211.045440.04.51	Програма та методика тестування	6	
5	A4	КПІ.ІП-5211.045440.05.34	Керівництво користувача	6	
6	A3	КПІ.ІП-5211.045440.06.99 СБД	Схема бази даних	1	
7	A3	КПІ.ІП-5211.045440.06.99 СС	Схема структурна класів програмного забезпечення	1	
8	A3	КПІ.ІП-5211.045440.06.99 СС	Схема структурна варіантів використань	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5211.045440.01.90		
Розробн.		Молявчик В.М.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-52	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 40 таблиць, 10 рисунків, 1 додаток та 10 джерел – загалом 73 сторінки.

Об`єкт дослідження: програмні комплекси, що агрегують на аналізують новинну повістку, використовуючи в якості джерел, як новинні сайти, так і соціальної мережі.

Мета дипломного проекту: розробити програмний продукт, що буде надавати набір інструментів для агрегації й аналізу новинної повістки, видаючи користувачеві завжди актуальну інформацію та оптимізувавши його час на її пошуки.

В рамках першого розділу був проведений огляд предметного середовища, були розглянені найближчі аналоги даної розробки, та визначені її завдання та вимоги.

В рамках другого розділу було змодельовано та сконструйовано основні аспекти та механізми програмнного забезпечення.

В рамках третього розділу було здійснено тестування розробленого застосунку. Були наведені деталі та опис пройдених тестів.

В рамках четвертого розділу були наведені всі необхідні інструкції для можливості запуску та роботи з застосунком.

В додатках наведені: лістинг програми.

КЛЮЧОВІ СЛОВА: АГРЕГАТОР, АНАЛІЗАТОР, ТРЕНДИ, НОВИНИ

ABSTRACT

Explanatory note of the diploma project consists of 4 sections, 40 tables, 10 illustrations, one annex and 10 sources – total 73 pages.

The object of study: program complexes that aggregate on the analysis of a news quiz, using as sources, both news sites and the social network.

The aim of the diploma project: to develop a software product that will provide a set of tools for aggregation and analysis of the informational space providing the user with always relevant information and optimizing time for quest.

In the context of the first section, the subject environment was conducted by a review. The nearest analogues of this development were considered, and its tasks and requirements were determined.

In the context of the second section, the main aspects and mechanisms of the software were modeled and constructed.

In the context of the third section, the developed application was performed by testing. The details and description of the passed tests were given.

In the context of the fourth section, all the necessary instructions were provided for setuping and running the application.

Annexes contain the listing of the program.

KEYWORDS: AGGREGATOR, ANALYZER, TRENDS, NEWS

Пояснювальна записка
до дипломного проекту

на тему: Комплекс задач агрегації та аналізу новин

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Веб-скрапінг – набір послідовних дій для зчитування та завантаження потрібної інформації, що розміщена на веб-сторінці та представлена однією з мов розмітки.

API – перелік програмних засобів, методів та стандартів, що дозволяють створення зв’язку між самостійними програмними компонентами в рамках однієї програми або створення такого зв’язку між окремими програмними комплексами.

RSS – мова розмітки з сімейства XML-подібних мов, що надає набір стандартизованих інструкцій для опису нових публікацій на сайті за затвердженою та визначеною структурою, що включає в себе практично всі дані конкретної новини.

Новинний агрегатор – програмний застосунок, функціонал якого полягає в автоматизованому збиранні новин з різних ресурсів та їх систематизації для користувача у зручному для нього вигляді та за визначеними критеріями.

Парсинг – набір інструкцій, що використовується для здійснення аналізу вхідних даних з подальшим приведенням вхідних даних до структури, що відповідає вказаній формальній граматиці.

ORM – сукупність методів, що створює проміжну ланку між базою даних та програмним комплексом, доволяючи використання сутностей бази даних за принципами об’єктно-орієнтованого програмування.

MVC – патерн архітектурного типу, за яким програмний комплекс розділяється на три взаємозалежні складові: модель відповідає за дані та операції з ними; представлення відповідає за відображення даних на частині клієнта; контролер відповідає за реагування на дії користувача.

ВСТУП

ХХІ століття стало справжньою інформаційною епохою в історії людства. Кожного дня ми читаємо публікації на новинних сайтах, дописи в соціальних мережах, отримуємо новини в месенджерах, ділимося та обговорюємо їх з друзями та знайомими. Тому новинні інтернет-ресурси та соціальні мережі стали відігравати значну роль в формуванні новинної повістки та розповсюдженні інформації, поступово витісняючи друковані видання та телевізійний формат подачі новин. Враховуючи це, традиційні джерела новин стали переходити в інтернет простір.

Зі збільшенням інформаційного простору з’явилася потреба в його агрегації, систематизації та аналізу для видачі користувачеві актуальної та важливої підбірки новин, що зроблена на основі найбільш популярних та авторитетних джерел інформації. Як вже було сказано, ітернет-простір постійно розширюється, тому розробка агрегаторів новин, які мають набір аналітичних інструментів є достатньо поширеною та впроваджує своє застосування на початкових сторінках пошукових порталів та систем або ж як додатки чи розширення на веб-сторінках, навіть не пов’язаних з новинною тематикою.

На даний момент відома достатня кількість прикладів вирішення подібних задач з реалізації новинних агрегаторів. Проте переважна більшість готових рішень має досить примітивний або ж взагалі відсутній функціонал для дослідження новинних трендів чи будь-яких інших показників. І лише одиниці підтримують агрегацію новинної повістки на основі стрічок соціальних мереж. До речі, в останньому випадку це зазвичай окремий продукт, що виконує агрегацію новин тільки з соціальних мереж, а не симбіозу новинних ресурсів та інтернет-спільнот. В якості прикладу роботи з інтернет-спільнотою було розглянуто соціальну мережу Twitter, що стала платформою для мережі мікроблогів на політичну та суспільну тематику.

Зважаючи на вище описані моменти, в рамках цієї дипломної роботи було розроблено програмний комплекс для агрегування та аналізу новин з

різнотипних джерел інформації, що також включають інтернет-спільности соціальних мереж.

В основі підтвердження актуальності даної розробки лежить постійне розширення інформаційного простору та необхідність в його агрегації та аналітиці засобами, що крокують в ногу з часом.

Основну ціль визначимо, як створення програмного комплексу, функціонал якого надає інструменти для агрегації та аналізу новинної повістки, сформованої на основі різнотипних джерел.

Метою розробки будемо вважати систематизацію різнопрофільних знань, що були отримані протягом всього навчального процесу та знадобляться для створення агрегатора та аналізатора новин із різнотипних ресурсів.

Призначенням розробки є подальше використання розробленого продукту, що дасть змогу користувачам оптимізувати свій час для отримання найбільш актуальних та важливих новин, як з найпопулярніших новинних ресурсів так і з соціальної мережі Twitter.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Сьогодні прогляднання публікацій на новинних ресурсах та стрічки новин соціальних мереж стало невід'ємною частиною нашого повсякденного життя. Інформаційний простір постійно розширюється великою кількістю інформації з різних, за форматом, способом подачі та змістом, ресурсів. ЗМІ та соціальні мережі практично моментально реагують на останні події, публікуючи актуальну інформацію. Проте, серед цих нескінченних об'ємів інформації, звичайному користувачеві досить складно виокремити головні новини, щоб залишатися в курсі останніх важливих подій.

Саме тому, все більшою популярністю користуються агрегатори новин, призначення котрих заключається в систематизації новинної повістки та видачі користувачеві стислої вибірки дійсно акутальних новин. Агрегатори ж з аналітичним функціоналом дають можливість слідкувати за трендами, зміною новинної тематики, як окремо взятих публікацій, так і повністю по обраному ресурсу протягом певного часового діапазону.

Ключова задача агрегатора – формування акутальної повістки та оптимізація часу користувача для її пошуку. Таким чином користувач отримує можливість залишатися в курсі останніх новин та подій, витративши на це мінмальні часові ресурси.

Ключовим моментом роботи таких програм є якнайшвидша обробка значних масивів інформації та виділення власними алгоритмами новин, які на даний час користуються популярністю, більшою за інші. Задача виділення популярних публікацій в певний момент часу грунтується не тільки на загальній кількості перегялядів, кількості поширень, дати пубілкації новини чи якихось інших мериках, а саме на динаміці зміни цих показників з плином часу.

Саме тому більшість агрегаторів працює в автономному режимі, оновлюючи свої бази інформації, та особливо динаміку змін по даних кожної зі

збережених публікації, на стільки часто, на стільки це можливо. Адже саме динаміка змін метрик публікацій дає можливість визначати її популярініть серед інших.

1.2 Змістовний опис і аналіз предметної області

1.2.1 Класифікація агрегаторів новин

Агрегатори новин за принципом роботи поділяються на два основні види: автоматичні та неавтоматичні агрегатори.

Автоматичні агрегатори працюють в режимі реального часу, та за автономним принципом, не допускаючи будь-якого втручання зі сторони.

Застосунки такого типу автоматично здійснюють підключення до джерел новин, це можуть бути інформаційні портали, агенства, електронні видання, новинні стрічки соціальних мереж. Після підключення до джерела, здійснюється збереженння даних, це не тільки новина та її заголовок, а практично всі по ній дані, наведемо список можливих показників та характеристик, що будуть використані в подальшому:

- кількість коментарів новини (якщо вони підтримуються);
- кількість поширень в соціальних мережах;
- кількість переглядів;
- заголовок новини;
- дата публікації;
- вкладені файли;
- текст новини;
- теги.

Вся ця інформація потрібна для алгоритмів, які визначають на скільки окрема взята новина популярна, формуючи в кінцевому результаті рейтинговий список новин за критерієм важливості та популярності окремо взятої новини на даний час.

Разом з тим алгоритми ранжування, яких використовують агрегатори,

можуть враховувати так званий «авторитет» ресурсу новини або ж його рейтинг серед інших, що формується на основі основних статистичних метрик попередніх новин ресурсу.

Неавтоматичні агрегатори працюють в ручному режимі, тобто новину повістку формує спеціальна команда фахівців, найчастіше журналістів, обираючи новини на власний розсуд.

Зрозуміло, що в таких умовах неавтоматичні агрегатори програють автоматичним, адже в цій ситуації грає людський фактор, неможливість навіть команди редакторів адекватно обробляти такі значні об’єми інформації та відсутність гарантій об’єктивності редакторського складу. Проте, неавтоматичні агрегатори можуть успішно використовуватись на тематичних сайтах або на головних сторінках ресурсів, як агрегатори новин виключно даного ресурсу, що пов’язано зі значно меншими масивами інформаційного навантаження та необхідністю в зацікавленні користувача авторських матеріалів.

Виокремимо основні параметри, якими користуються новинні агрегатори:

- актуальність новини, визначається відносно дати публікації;
- статистичні показники новини;
- динаміка росту статистичних показників;
- рейтинг (популярність).

1.2.2 Правовий статус новинних агрегаторів

На сьогоднішній день правовий статус новинних агрегаторів жодним чином не врегульовано. Через що в європейських правових інституціях тривають обговорення про зобов’язання агрегаторів новин перед новинними ресурсами, зокрема здійснення виплат за розміщення їх публікацій та збереження авторських прав, шляхом вказання та повноцінного посилання на використані агрегатором джерела. Проте вирішення даного питання поки що зупинилося лише на етапі обговорень, саме тому зараз агрегатори мають цілком

законне право використовувати новинні ресурси, залишаючи на них посилання.

1.2.3 Інтеграція з соціальними мережами

В середовищі новинних агрегаторів були випадки конвергенції з соціальними мережами. Наприклад, в 2016 році Рамблер/Новини почали інтеграцію з соціальною мережею «ВКонтакте», включаючи дописи з сторінок соцмережі в інформаційний блок новин.

Конвергенція була узгоджена договором про співпрацю з мережею, в свою чергу Рамблер/Новини самостійно агрегували дописи соціальною мережі, без її втручання. Станом на 2019 рік співпрацю припинено, офіційні причин зупинення інтеграції не відомі.

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

Технічні рішення що використовуються в розглянутій предметній області базуються на принципах надійності, швикодії та безпеки даних. Процес веб-скрапінгу публікацій лежить в основі механізму роботи програми. Розглянемо інструменти, що для цього використовуються.

Selenium – портативний фреймворк, що надає набір інструментів для автоматизованого виконання задач в браузері. Зокрема Selenium Webdriver в автоматичному режимі запускає команди в браузері з можливістю отримання повноцінного доступу до додатку.

Scrapy – веб-фреймворк мови програмування Python, що вже має напівготову інфраструктуру для здійснення веб-скрапінгу за допомогою автономних компонетів, що називаються «павуки», та приймають список, необхідних для скрапінгу, параметрів.

Beautiful Soup 4 – бібліотека, написана на мові програмування Python, що сторена спеціально для детального скрапінгу веб-сторінок, з широким

інструментарієм для зручного виокремлення інформації з структурованих даних.

1.3.2 Аналіз відомих програмних продуктів

Нарешті, перейдімо до розгляду успішних продуктів, які є найближчими аналогами до розробленого проекту.

Найбільш близьким за функціоналом до недавнього часу був український продукт, новинний агрегатор – Zmiya. Проте станом на 2019 рік, він за невідомих причин призупинив свою роботу. Сервіс в режимі реального часу агрегував інформацію, як з новиних сайтів, так і з сторінок та блогів соціальних мереж. Платформа надавала окремі сторінки для аналізу ресурсів, профілів, можливістю слідкувати за кількістю підписників, кількістю поширень в соціальних мережах, кількістю відміток «мені подобається», кількістю коментарів, та динамікою змін кожного з перерахованих статистичних показників. За цими даними платформа формувала рейтингові списки публікаторів. Також застосунок підтримував свою власну мережу мікроблогів, тому кожен зареєстрований користувач мав шанс потрапити в топ новинної повістки опублікувавши авторський матеріал.

Наступний кандидат – Яндекс.Новини, цей сервіс вже не виступає аналізатором, лише агрегатором, адже не має жодних аналітичних інструментів. Проте має хороші агрегаційні можливості, включаючи можливість отримувати новинну повістку обраної країни або ж навіть окремого регіону країни. Також важливо зазначити, що сервіс підтримує відмінне розділення публікацій за категоріями та темами. Продукт орієнтований на пострадянський простір. У зв’язку з українсько-російським конфліктом був заблокований на території України рішенням РНБО.

Український аналог попереднього прикладу – ukr.net. Даний веб-застосунок практично повторює функціонал попередника. За вийнятком локалізації новин по територіальному признаку, тут розміщуються тільки

українські новини. Інтерфейс трохи застрілий, так як практично не змінювався з моменту створення. Сервіс працює на українському інфопросторі ще з далекого 1998 року.

Останнім повноцінним аналогом розглянемо новинний агрегатор від корпорації Google – Google Новини. Сервіс підтримує локалізацію за територіальним призником та класифікацію новин за категоріями. Також на відміну від всіх попередників агрегатор дає можливості отримувати персоналізовану новинну повістку, тобто виходячи з попередніх інтересів користувача.

Проблеми швидкості та актуальності для агрегаторів по суті є питаннями про їх життя, чи знайде агрегатор свою аудиторію, чи вийде він на рівень країни. Перелічені вище ресурси користуються або користувалися досить широким попитом, як в Україні так і за кордоном. Проте тільки один з них мав повноціннний аналітичний функціонал – Zmiya, та й той призупинив свою роботу.

1.4 Аналіз вимог до програмного забезпечення

Застосунок має підтримувати дві функціональні ролі: роль адміністратора та роль користувача.

Адміністратором вважається користувач, який має певні привілегії, зокрема через програмний інтерфейс вносити будь-які зміни в базу даних комплексу.

Користувачем же вважається потенційний клієнт розробленого продукту, перед ним відкритий повний функціонал застосунку: починаючи з можливості перегляду агрегованих списків новин та закінчуючи завантаженням зображення побудованих графіків трендів новин у форматі PNG.

Виокремимо дві основні лінії з яких складається програмне забезпечення:

- агрегація списків новин за певними критеріями;
- аналітичні й оптимізаційні методи обробки збережених даних.

1.4.1 Розроблення функціональних вимог

Схема структурна варіантів використання наведена у графічному матеріалі. Наступні ж таблиці 1.1 - 1.12 представляють функціональні вимоги описані варіантами використання.

Таблиця 1.1 – Варіант використання UC001

Назва	Збереження графіку трендів блогів Twitter на ПК
Опис	Користувач спроможний зберегти графічно візуалізовані тренди блогів Twitter
Учасники	Користувач
Передумови	Відкрита сторінка аналітики застосунку, графіки трендів блогів Twitter побудовані
Постумови	Користувач завантажує графіки у форматі зображення, що відображають тренди блогів Twitter
Основний сценарій	Користувач обирає необхідний побудований графік трендів блогів Twitter та натискає навпроти нього кнопку «Зберегти» Застосунок завантажує графік трендів блогів Twitter в форматі PNG
Розширення сценаріїв	

Таблиця 1.2 – Варіант використання UC002

Назва	Внесення даних
Опис	Адміністратор спроможний вносити дані по будь-якій сутності в базу даних
Учасники	Адміністратор
Передумови	Здійснено авторизацію адміністратора
Постумови	Дані по обраній сутності внесені в базу даних
Основний сценарій	Адміністратор переходить в «Адміністрація сайту»

Продовження таблиці 1.2

	Застосунок виводить на сторінку список всіх сутностей Адміністратор обирає сутність та натискає кнопку «Додати» навпроти неї Застосунок виводить на сторінку форму для обраної сутності Відповідні поля форми заповнюються адміністратором Далі адміністратором натискається кнопка «Зберегти» Застосунок здійснює збереження даних по обраній сутності
Розширення сценаріїв	Застосунок не здійснює збереження даних по обраній сутності через некоректно введені дані Застосунок відображає помилку введення даних

Таблиця 1.3 – Варіант використання UC003

Назва	Видалення даних
Опис	Адміністратор спроможний видалити дані по будь-якій сутності з бази даних
Учасники	Адміністратор
Передумови	Здійснено авторизацію адміністратора
Постумови	Дані по обраній сутності видалені з бази даних
Основний сценарій	Адміністратор переходить в розділ «Адміністрація сайту» Застосунок виводить на сторінку список всіх сутностей Адміністратор обирає сутність та натискає кнопку «Змінити» навпроти неї Застосунок виводить на сторінку список всіх об’єктів обраної сутності Адміністратор обирає запис сутності

Продовження таблиці 1.3

	Адміністратор обирає запис сутності Застосунок виводить на сторінку форму для обраної сутності Далі адміністратором натискається кнопка «Видалити» Застосунок здійснює видалення даних по обраній сутності
Розширення сценаріїв	

Таблиця 1.4 – Варіант використання UC004

Назва	Редагування даних
Опис	Адміністратор спроможний редагувати дані по будь-якій сутності в базі даних
Учасники	Адміністратор
Передумови	Здійснено авторизацію адміністратора
Постумови	Дані по обраній сутності відредаговані та збережені в базу даних
Основний сценарій	Адміністратор переходить в розділ «Адміністрація сайту» Застосунок виводить на сторінку список всіх сутностей Адміністратор обирає сутність та натискає кнопку «Змінити» навпроти неї Застосунок виводить на сторінку список всіх об’єктів обраної сутності Адміністратор обирає запис сутності Застосунок виводить на сторінку форму для обраної сутності Відповідні поля форми редагуються адміністратором Далі адміністратором натискається кнопка «Зберегти»

Продовження таблиці 1.4

	Застосунок здійснює збереження даних по обраній сутності
Розширення сценаріїв	Застосунок не здійснює збереження даних по обраній сутності через некоректно введені дані Застосунок відображає помилку введення даних

Таблиця 1.5 – Варіант використання UC005

Назва	Переглядання агрегованого списку новин
Опис	Користувач спроможний отримати повний список останніх або найпопулярніших, або актуальних публікацій новинних сайтів
Учасники	Користувач
Передумови	Відкрита головна сторінка застосунку
Постумови	Користувач перенаправляється на сторінку з повним списком останніх або найпопулярніших, або актуальних публікацій новинних сайтів
Основний сценарій	Користувач натискає кнопку «Усі новини» у відповідній категорії сайту: «Останні» або «Найпопулярніші», або «Актуальні» Застосунок відкриває сторінку відповідної категорії та виводить на екран запитуваний список публікацій
Розширення сценаріїв	

Таблиця 1.6 – Варіант використання UC006

Назва	Переглядання рейтингового списку новинних ресурсів
Опис	Користувач спроможний отримати рейтинговий список новинних ресурсів, що складається на основі статистичних показників

Продовження таблиці 1.6

Учасники	Користувач
Передумови	Відкрита сторінка статистики застосунку
Постумови	Користувач отримує статистичний рейтинговий список новинних ресурсів
Основний сценарій	Користувач обирає часовий діапазон, натиснувши на одну з кнопок: «За добу» або за «За тиждень» Застосунок виводить на екран рейтинговий список новиних ресурсів за вибраний часовий діапазон
Розширення сценаріїв	

Таблиця 1.7 – Варіант використання UC007

Назва	Графічне предсталення аналітичних метрик новинних ресурсів
Опис	Користувач спроможний отримати графічно візуалізовані тренди новинних ресурсів
Учасники	Користувач
Передумови	Відкрита сторінка аналітики застосунку
Постумови	Користувач отримує графіки, що графічно відображають тренди новинних ресурсів
Основний сценарій	Користувач обирає часовий діапазон, натиснувши на одну з кнопок: «За добу» або за «За тиждень» Користувач обирає новинний ресурс, натиснувши на відповідну йому кнопку Застосунок виводить на екран графіки трендів вибраного новинного ресурсу за відповідний часовий діапазон
Розширення сценаріїв	

Таблиця 1.8 – Варіант використання UC008

Назва	Збереження графіку трендів новинних сайтів на ПК
Опис	Користувач спроможний зберегти графічно візуалізовані тренди новинних ресурсів
Учасники	Користувач
Передумови	Відкрита сторінка аналітики застосунку, графіки трендів новинних сайтів побудовані
Постумови	Користувач завантажує графіки у форматі зображення, що відображають тренди новинних ресурсів
Основний сценарій	Користувач обирає необхідний побудований графік трендів новинних ресурсів та натискає навпроти нього кнопку «Зберегти» Застосунок завантажує графік трендів новинних сайтів в форматі PNG
Розширення сценаріїв	

Таблиця 1.9 – Варіант використання UC009

Назва	Переглядання агрегованого списку дописів
Опис	Користувач спроможний отримати повний список останніх або найпопулярніших, або актуальних дописів соціальної мережі Twitter
Учасники	Користувач
Передумови	Відкрита головна сторінка застосунку
Постумови	Користувач перенаправляється на сторінку з повним списком останніх або найпопулярніших, або актуальних дописів соціальної мережі Twitter
Основний сценарій	Користувач натискає кнопку «Усі дописи» у відповідній категорії сайту: «Останні» або «Найпопулярніші», або «Актуальні»

Продовження таблиці 1.9

	Застосунок відкриває сторінку відповідної категорії та виводить на екран запитуваний список дописів
Розширення сценаріїв	

Таблиця 1.10 – Варіант використання UC010

Назва	Переглядання рейтингового списку блогів Twitter
Опис	Користувач спроможний отримати рейтинговий список блогів соціальної мережі Twitter, що складається на основі статистичних показників
Учасники	Користувач
Передумови	Відкрита сторінка статистики застосунку
Постумови	Користувач отримує статистичний рейтинговий список блогів соціальної мережі Twitter
Основний сценарій	Користувач обирає часовий діапазон, натиснувши на одну з кнопок: «За добу» або за «За тиждень» Застосунок виводить на екран рейтинговий список блогів соціальної мережі Twitter за вибраний часовий діапазон
Розширення сценаріїв	

Таблиця 1.11 – Варіант використання UC011

Назва	Графічне предсталення аналітичних метрик блогів Twitter
Опис	Користувач спроможний отримати графічно візуалізовані тренди блогів Twitter
Учасники	Користувач
Передумови	Відкрита сторінка аналітики застосунку
Постумови	Користувач отримує графіки, що графічно

Продовження таблиці 1.11

	відображають тренди блогів Twitter
Основний сценарій	Користувач обирає часовий діапазон, натиснувши на одну з кнопок: «За добу» або за «За тиждень» Користувач обирає блог Twitter, натиснувши на відповідну йому кнопку Застосунок виводить на екран графіки трендів вибраного блогу Twitter за відповідний часовий діапазон
Розширення сценаріїв	

Таблиця 1.12 – Варіант використання UC012

Назва	Авторизація адміністратора
Опис	Адміністратор спроможний здійснити авторизацію в застосунку
Учасники	Адміністратор
Передумови	Відкрита сторінка авторизації
Постумови	Здійснено авторизацію адміністратора
Основний сценарій	Застосунок відображає поля для авторизації (електронна пошта та пароль) Відповідні поля заповнюються адміністратором Далі адміністратором натискається кнопка «Вхід» Застосунок здійснює авторизацію адміністратора
Розширення сценаріїв	Застосунок не здійснює авторизацію адміністратора через неправильно введені дані Застосунок відображає помилку авторизації

Наступні таблиці 1.13 - 1.33 представлені описом функціональних вимог до розроблюваного застосунку.

Таблиця 1.13 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Підтримка внесення даних
Опис	Застосунок повинен підтримувати спроможність внесення даних по будь-якій сутності в базу даних

Таблиця 1.14 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Підтримка видалення даних
Опис	Застосунок повинен підтримувати спроможність видалення даних по будь-якій сутності з бази даних

Таблиця 1.15 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Підтримка оновлення даних
Опис	Застосунок повинен підтримувати спроможність оновлення даних по будь-якій сутності в базі даних

Таблиця 1.16 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Переглядання списку актуальних новин
Опис	Здійснивши запит на отримання всіх новин з категорії «Актуальні», користувач має змогу переглядати список всіх актуальних публікацій новинних сайтів, підібраних за спеціальним алгоритмом

Таблиця 1.17 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Переглядання списку останніх новин

Продовження таблиці 1.17

Опис	Здійснивши запит на отримання всіх новин з категорії «Останні», користувач має змогу переглядати список всіх останніх публікацій новинних сайтів, відсортованих за датою публікації

Таблиця 1.18 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Переглядання списку найпопулярніших новин
Опис	Здійснивши запит на отримання всіх новин з категорії «Найпопулярніші», користувач має змогу переглядати список всіх найпопулярніших публікацій новинних сайтів, відсортованих за кількістю переглядів

Таблиця 1.19 – Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Вибір часового проміжку для статистики по ЗМІ
Опис	Користувач має змогу обирати часовий проміжок для якого будуть зібрані статистичні дані про новинні ресурси

Таблиця 1.20 – Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Вибір часового проміжку для статистики по Twitter
Опис	Користувач має змогу обирати часовий проміжок для якого будуть зібрані статистичні дані про блоги мережі Twitter

Таблиця 1.21 – Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Переглядання статистики по ЗМІ

Продовження таблиці 1.21

Опис	Користувач має змогу переглядати статистичні показники новинних ресурсів за обраним часовим проміжком

Таблиця 1.22 – Опис функціональної вимоги REQ010

Номер	REQ010
Назва	Переглядання статистики по Twitter
Опис	Користувач має змогу переглядати статистичні показники блогів Twitter за обраним часовим проміжком

Таблиця 1.23 – Опис функціональної вимоги REQ011

Номер	REQ011
Назва	Графічна візуалізація трендів ЗМІ
Опис	Обравши новинний ресурс та часовий проміжок, користувач має змогу переглянути графічну візуалізацію знайдених трендів

Таблиця 1.24 – Опис функціональної вимоги REQ012

Номер	REQ012
Назва	Графічна візуалізація трендів Twitter
Опис	Обравши блог мережі Twitter та часовий проміжок, користувач має змогу переглянути графічну візуалізацію знайдених трендів

Таблиця 1.25 – Опис функціональної вимоги REQ013

Номер	REQ013
Назва	Вибір часового проміжку для дослідження трендів у ЗМІ
Опис	Користувач має змогу обрати часовий проміжок для якого будуть знайдені тренди вказаних новинних ресурсів

Таблиця 1.26 – Опис функціональної вимоги REQ014

Номер	REQ014
Назва	Вибір часового проміжку для дослідження трендів у Twitter
Опис	Користувач має змогу обрати часовий проміжок для якого будуть знайдені тренди вказаних блогів мережі Twitter

Таблиця 1.27 – Опис функціональної вимоги REQ015

Номер	REQ015
Назва	Вибір новинних ресурсів для дослідження їх трендів
Опис	Користувач має змогу обрати новинний ресурс для якого й будуть знайдені тренди

Таблиця 1.28 – Опис функціональної вимоги REQ016

Номер	REQ016
Назва	Вибір блогів Twitter для дослідження їх трендів
Опис	Користувач має змогу обрати блог Twitter для якого й будуть знайдені тренди

Таблиця 1.29 – Опис функціональної вимоги REQ017

Номер	REQ017
Назва	Переглядання списку актуальних дописів Twitter
Опис	Здійснивши запит на отримання всіх дописів з категорії «Актуальні», користувач має змогу переглядати список всіх актуальних дописів Twitter, підібраних за спеціальним алгоритмом

Таблиця 1.30 – Опис функціональної вимоги REQ018

Номер	REQ018
Назва	Переглядання списку останніх дописів Twitter

Продовження таблиці 1.31

Опис	Здійснивши запит на отримання всіх дописів з категорії «Останні», користувач має змогу переглядати список всіх останніх дописів Twitter, відсортованих за датою публікації

Таблиця 1.31 – Опис функціональної вимоги REQ019

Номер	REQ019
Назва	Переглядання списку найпопулярніших дописів Twitter
Опис	Здійснивши запит на отримання всіх дописів з категорії «Найпопулярніші», користувач має змогу переглядати список всіх найпопулярніших дописів Twitter, відсортованих за кількістю переглядів

Таблиця 1.32 – Опис функціональної вимоги REQ020

Номер	REQ020
Назва	Збереження графіку
Опис	Застосунок повинен підтримувати збереження вибраних коритсувачем графіків у форматі PNG

Таблиця 1.33 – Опис функціональної вимоги REQ021

Номер	REQ021
Назва	Підтримка авторизації в застосунку
Опис	Застосунок повинен тримати збереженими особисті дані адміністратора та підтримувати спроможність авторизації за їх допомогою

Зв’язок між описаними вище вимогами REQXXX та варіантами UCXXX проілюструємо рисунком 1.1.

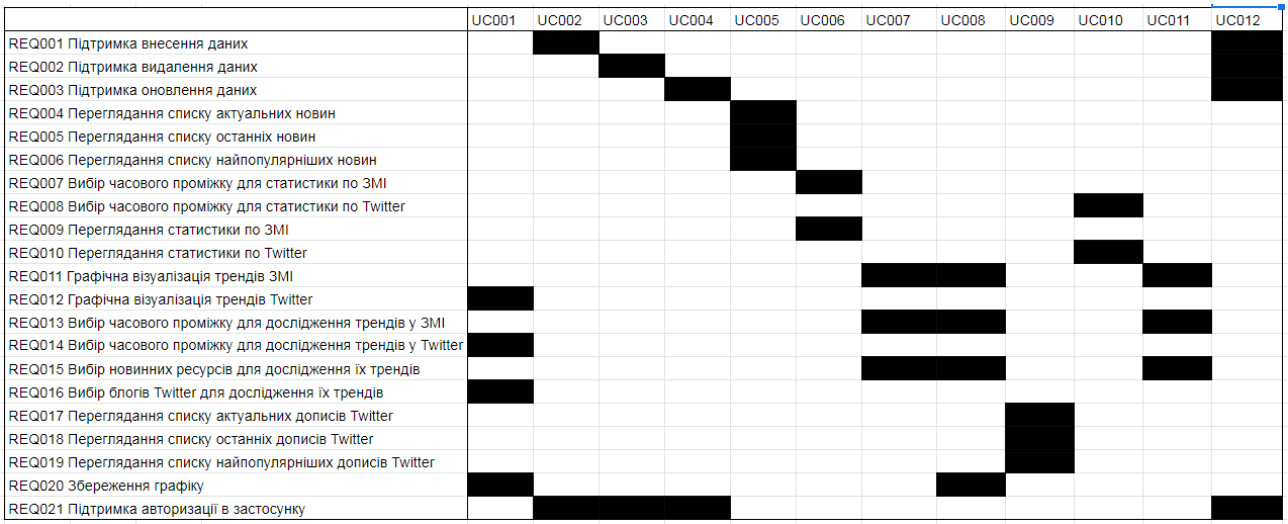

Рисунок 1.1 – Матриця відповідності вимог

1.4.2 Розроблення нефункціональних вимг

Вимоги до апаратного забезпечення банально мінімальні – практично будь-який комп’ютер з можливістю підключення до Інтернету;

Вимоги до до програмного забезпечення – ОС Linux Ubuntu, починаючи з версії 14.4 та Mac OS починаючи з версії 10.14.5.

Вимоги до GUI – веб-застосунок з можливістю відтворення зрозумілих та чітких діаграм на основі переданих даних.

Вимоги до бази даних – повинна використовуватися СКБД, що успішно вміє працювати з великими навантаженнями, наприклад, PostgreSQL.

1.4.3 Постановка комплексу завдань модулю

Основна ціль була визначена створенням програмного комплексу, функціонал якого надає інструменти для здійснення агрегації новинної повістки, сформованої на основі різнотипних джерел, а також проведення над нею відповідних аналітичних заходів.

Відносно поставленої цілі можна виокремити самостійні компоненти застосунку, що будуть формувати цілісну картину продукту:

- веб-скрапінг новинних ресурсів;
- отримання даних з Twitter через API;

- збереження публікацій та дописів до бази даних;
- збереження інформації про ресурси та блоги в базу даних;
- синтаксичний та семантичний розбір новин;
- виокремлення трендів кожної із збережених новин;
- видача користувачеві агрегованої новинної повістки;
- видача користувачеві статистики по джерелах новин;
- видача користувачеві графічних елементів, що відображають тренди.

1.5 Висновки по розділу

В рамках першого розділу було досліджено предметну область дипломної роботи, за допомогою її змістовного опису та аналізу. Були розглянуті переваги та недоліки найближчих аналогів розробки.

Були проаналізовані загальновідомі технічні рішення, що використовуються в розробках такої спеціалізації.

Були сформульовані як функціональні, так і нефункціональні вмоги, якими потрібно керуватися при розробці програми, план реалізації якої було розбито на автономні компоненти під час поставлення завдань.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для здійснення процесу моделювання виокремимо головні бізнес-процеси комплексу з вирішення завдань агрегації й аналізу новинної повістки, спираючись на різнотипні новинні джерела, що включають не тільки новинні сайти, а й мікроблоги соціальних мереж.

Отже, визначимо ж головні бізнес-процеси даного застосунку:

– отримання повного списку агрегованих новин за одним з критеріїв;

– отримання рейтингового списку ресурсів новин на основі їхніх статистичних метрик за вибраний часовий діапазон;

– визначення новинних трендів та подальша їх візуалізація по обраному ресурсу новин за вибраний часовий діапазон.

Приступимо до побудови діаграм виокремлених бізнес-процесів та детально опишемо кожен з них.

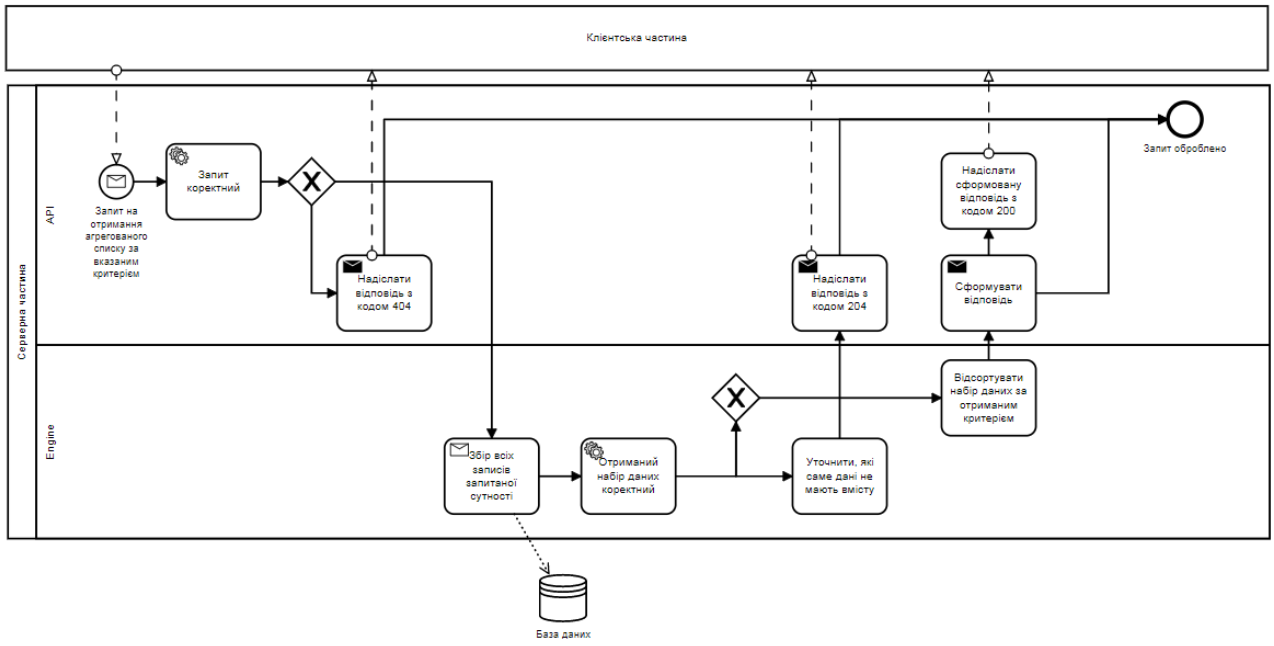

Рисунок 2.1 – Діаграма бізнес-процесу обробки запиту за критерієм

Перед нами схема бізнес-процесу обробки запиту, за яким серверна частина повинна повернути на клієнтську агрегований список новин за переданим параметром, керуючись якого буде здійснена агрегація новинної повістки.

Спочатку перевіряється коректність запиту чи підтримує API запитуваний метод та чи всі параметри були коректно передані. Далі відбувається комунікація з базою даних протягом якої отримується набір об’ктів запитуваної сутності. Отримавши цей набір перевіряється його коректність і вразі помилки, відправляється відповідь з кодом помилки.

Далі отриманий набір даних сортується або ж ранжується за алгоритмом, це все в залежності від переданого критерію новинної повістки. Опісля формується відповідь з агрегованим списком даних та успішним кодом 200. Цей етап є завершальним для розглянутого бізнес-процесу.

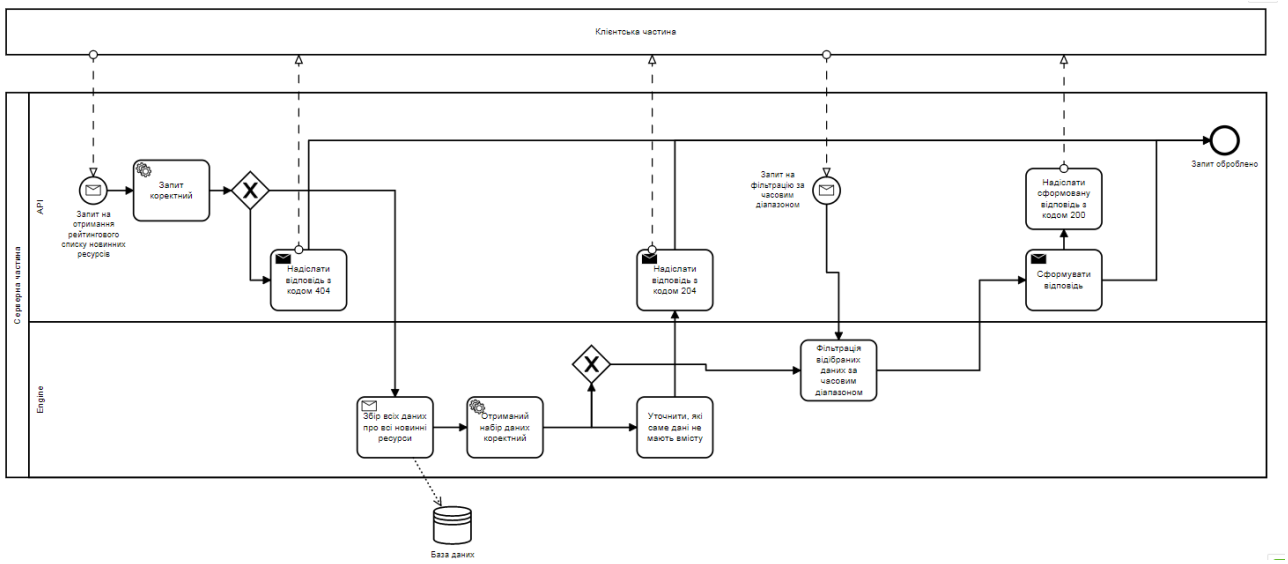

Рисунок 2.2 – Діаграма бізнес-процесу обробки запиту на отримання статистики

Перед нами схема бізнес-процесу обробки запиту, за яким серверна частина повинна повернути на клієнтську рейтинговий список новинних ресурсів за вказаний діапазон часу. Спочатку перевіряється коректність запиту чи підтримує API запитуваний метод та чи всі параметри були коректно передані. Далі відбувається комунікація з базою даних протягом якої

отримується набір об’ктів запитуваної сутності. Отримавши цей набір перевіряється його коректність і вразі помилки, відправляється відповідь з кодом помилки.

Далі отриманий набір даних фільтрується за переданим часовим діапазоном та ранжується за статистичними метриками кожного з новинних ресурсів. Опісля формується відповідь з рейтинговим списком даних та успішним кодом 200. Цей етап є завершальним для розглянутого бізнес-процесу.

2.2 Архітектура програмного забезпечення

Розроблений комплекс призначений для виконання на UNIX-подібних операційних система, зокрема це можуть бути будь-які дистрибутиви Linux.

Клієнтська частина застосунку була написана з використанням Bootstrap, AJAX, jQuery. Для реалізації серверної частини було обрано мову програмування Python, зокрема її веб фреймворк для розробки веб рішень – Django.

В якості шаблону проектування було вирішено використати архітектурний паттерн – MVC. Цей патерн розділяє систему на три незалежні, проте пов’язані частини: модель даних, представлення даних та оперування даними. Модель даних в цьому принципі відповідає за збереження даних від можливого вручання. Представлення даних дає зручні шаблони за допомогою, яких можна легко відображати значні масиви інформації на сторінці. Оперування даними – по суті проміжна ланка між попередніми двома компонентами, допомагає правильно змінювати та отримувати дані.

В якості бази даних була задіяна об'єктно-реляційна система керування базами даних PostgreSQL, що перш за все славиться своєю надійністю та стабільністю роботи в екстремальних умовах.

Опис класів програми наведено у таблиці 2.1, опис методів класів – у таблиці 2.2, опис структури бази даних – таблиця 2.3.

2.3 Конструювання програмного забезпечення

Діаграма класів програмного комплексу наведена у графічному матеріалі. Детальний опис класів та їх методів наведений нижче в таблицях.

Таблиця 2.1 – Опис класів комплексу

Клас	Опис
Connector	Клас, функціонал якого полягає в підключенні до API Twitter
Parser	Клас, функціонал якого полягає в налаштуванні бібліотек для скрапінгу веб-сторінок
NewsParser (Parser)	Реалізовує скрапінг новин з сайту
TwitterParser (Connector)	Реалізовує отримання твітів з соціальної мережі Twitter через API
NewsProfileParser (Parser)	Реалізовує скрапінг інформації про новинний ресурс
TwitterProfileParser (Connector)	Реалізовує отримання інформації про сторінку соціальної мережі Twitter через API
DBDataSaver	Надає функціонал для збереження даних в базу даних
PreviousAnalyzer	Реалізовує функціонал для попереднього аналізу текстів та відкидання стоп-слів
Word	Клас-модель бази даних для зберігання токенів
TwitterWordCount	Клас-модель бази даних для зберігання статистики по токенах (Twitter)

Продовження таблиці 2.1

NewsWordCount	Клас-модель бази даних для зберігання статистики по токенах (новинні ресурси)
ArticleItems	Клас-модель бази даних для зберігання новинних статей
TweetItems	Клас-модель бази даних для зберігання твітів з Twitter
SiteProfile	Клас-модель бази даних для зберігання інформації про новинний ресурс
TwitterProfile	Клас-модель бази даних для зберігання інформації про профільTwitter
ArticleItemsSerializer	Клас-серіалізатор, що дозволяє перетворити querysets статтей в формат JSON
TweetItemsSerializer	Клас-серіалізатор, що дозволяє перетворити querysets твітів в формат JSON

Таблиця 2.2 – Опис методів класів комплексу

Клас	Метод	Опис
Connector	__init__	Конструктор, ініціалізація початкових значень атрибутів
Connector	create_connection	Створює з’єдання з Twitter API
Connector	check_connection	Відповідає за перевірку з’єднання з Twitter API
Parser	__ init__	Конструктор, ініціалізація початкових значень

Продовження таблиці 2.2

		атрибутів
NewsParser (Parser)	__ init__	Конструктор, ініціалізація початкових значень атрибутів
NewsParser (Parser)	get_data	Скрапінг сторінки та отримання її в форматі XML
NewsParser (Parser)	get_title	Витяг заголовку статті з набору даних
NewsParser (Parser)	get_published	Витяг дати публікації статті з набору даних
NewsParser (Parser)	get_link	Витяг посилання на статтю з набору даних
NewsParser (Parser)	get_text	Витяг тексту статті з набору даних
NewsParser (Parser)	get_views	Витяг кількості переглядів статті з набору даних
TwitterParser (Connector)	__ init__	Конструктор, ініціалізація початкових значень атрибутів
TwitterParser (Connector)	get_comments_count	Витяг кількості коменарів допису з набору даних
TwitterParser (Connector)	get_likes_count	Витяг кількості лайків допису з набору даних
TwitterParser (Connector)	get_retweets _count	Витяг кількості репостів допису з набору даних
NewsProfileParser	__ init__	Конструктор, ініціалізація початкових значень

Продовження таблиці 2.2

		атрибутів
NewsProfileParser	get_link	Витяг посилання на новинний ресурс з набору даних
NewsProfileParser	get_rss_link	Витяг посилання на rss-розсилку новинного ресурсу з набору даних
NewsProfileParser	get_logo	Витяг лого новинного ресурсу з набору даних
NewsProfileParser	get_name	Витяг назви новинного ресурсу з набору даних
TwitterProfileParser	__ init__	Конструктор, ініціалізація початкових значень атрибутів
TwitterProfileParser	get_link	Витяг посилання на сторінку Twitter з набору даних
TwitterProfileParser	get_name	Витяг назви сторінки Twitter з набору даних
TwitterProfileParser	get_username	Витяг юзернейма сторінки Twitter з набору даних
TwitterProfileParser	get_logo	Витяг лого сторінки Twitter з набору даних
TwitterProfileParser	get_subscribers_count	Витяг кількості підписників сторінки Twitter з набору даних
DBDataSaver	__ init__	Конструктор, ініціалізація початкових значень

Продовження таблиці 2.2

		атрибутів
DBDataSaver	save_news	Реалізує збереження новинної статті до бази даних
DBDataSaver	save_tweet	Реалізує збереження допису Twitter до бази даних
DBDataSaver	save_twitter_wordcount	Реалізує збереженння статистичних даних допису Twitter до бази даних
DBDataSaver	save_news_wordcount	Реалізує збереженння статистичних даних статті новинного ресурсу до бази даних
DBDataSaver	save_word	Реалізує збереженння токену до бази даних
PreviousAnalyzer	__ init__	Конструктор, ініціалізація початкових значень атрибутів
PreviousAnalyzer	get_stopwords_list	Отримання списку стоп-слів
PreviousAnalyzer	tokenize	Формування токенів з вхідного тексту
PreviousAnalyzer	clear_text_of_stop_words	Очищення вхідного тексту від стоп-слів
PreviousAnalyzer	word_count	Підрахунок кількості входжень кожного з токенів у вхідний текст
PreviousAnalyzer	word_count_diff	Обчислити метрики

Продовження таблиці 2.2

		кожного з токенів
TweetItems	__str__	Реалізує відображення об'єкта класу TweetItems на сайті адміністратора
TweetItems	get_absolute_url	Формує канонічну url-адресу для об’єкта класу TweetItems
TweetItems	save	Збереження в базу даних об’єкта класу TweetItems
ArticleItems	__str__	Реалізує відображення об'єкта класу ArticleItems на сайті адміністратора
ArticleItems	get_absolute_url	Формує канонічну url-адресу для об’єкта класу ArticleItems
ArticleItems	save	Збереження в базу даних об’єкта класу ArticleItems
SiteProfile	__str__	Реалізує відображення об'єкта класу SiteProfile на сайті адміністратора
SiteProfile	get_absolute_url	Формує канонічну url-адресу для об’єкта класу SiteProfile
SiteProfile	save	Збереження в базу даних об’єкта класу SiteProfile
TwitterProfile	__str__	Реалізує відображення об'єкта класу TwitterProfile на сайті адміністратора

Продовження таблиці 2.2

TwitterProfile	get_absolute_url	Формує канонічну url-адресу для об’єкта класу TwitterProfile
TwitterProfile	save	Збереження в базу даних об’єкта класу TwitterProfile
ArticleItemsSerializer	create	Створення повного екземпляра об’єкта класу ArticleItemsSerializer на основі перевірених даних
ArticleItemsSerializer	update	Редагування повного екземпляра об’єкта класу ArticleItemsSerializer на основі перевірених даних
ArticleItemsSerializer	validate	Виконує валідацію об’єкта класу ArticleItemsSerializer
ArticleItemsSerializer	to_representation	Створення словника атрибутів об’єкта класу ArticleItemsSerializer для передачі на клієнтську частину
ArticleItemsSerializer	__init__	Конструктор, ініціалізація початкових значень атрибутів
TweetItemsSerializer	create	Створення повного екземпляра об’єкта класу TweetItemsSerializer на основі перевірених даних
TweetItemsSerializer	update	Редагування повного

Продовження таблиці 2.2

		екземпляра об’єкта класу ArticleItemsSerializer на основі перевірених даних
TweetItemsSerializer	validate	Виконує валідацію об’єкта класу TweetItemsSerializer
TweetItemsSerializer	to_representation	Створення словника атрибутів об’єкта класу TweetItemsSerializer для передачі на клієнтську частину
TweetItemsSerializer	__init__	Конструктор, ініціалізація початкових значень атрибутів

Таблиця 2.3 – Опис таблиць бази даних комплексу

Таблиця	Запис	Тип змінної	Опис
NewsSite	id (primary key)	integer	Ідентифікатор новинного сайту
NewsSite	bio	varchar(1000)	Коротка інформація про новинний сайт
NewsSite	username	varchar(100)	Скорочена назва новинного сайту
NewsSite	fullname	varchar(200)	Повна назва новинного сайту
NewsSite	link	varchar(200)	Посилання на новинний сайт
NewsSite	rsslink	varchar(400)	Посилання на rss-розсилку новинного

			сайту
NewsSite	logo	varchar(400)	Лого новинного сайту
TwitterProfile	id (primary key)	integer	Ідентифікатор сторінки мережі Twitter
TwitterProfile	fullname	varchar(200)	Назва сторінки мережі Twitter
TwitterProfile	username	varchar(100)	Юзернейм сторінки мережі Twitter
TwitterProfile	link	varchar(200)	Посилання на сторінку мережі Twitter
TwitterProfile	logo	varchar(400)	Лого сторінки мережі Twitter
TwitterProfile	subscribers	integer	Кількість підписників сторінки мережі Twitter
TwitterProfile	bio	varchar(1000)	Коротка інформація про сторінку мережі Twitter
Article	id (primary key)	integer	Ідентифікатор статті новинного сайту
Article	title	varchar(500)	Текст заголовку статті новинного сайту
Article	published	date	Дата публікації статті новинного сайту
Article	link	varchar(500)	Посилання на статтю новинного сайту
Article	text	varchar(10000)	Текст статті новинного сайту
Article	site_id (foreign key)	integer	Ідентифікатор новинного сайту

Article	views	integer	Кількість переглядів статті новинного сайту
Article	comments	integer	Кількість коментарів статті новинного сайту
Tweet	id (primary key)	integer	Ідентифікатор допису мережі Twitter
Tweet	comments	integer	Кількість коментарів допису мережі Twitter
Tweet	retweets	integer	Кількість репостів допису мережі Twitter
Tweet	likes	integer	Кількість вподобань допису мережі Twitter
Tweet	twitter_id (foreign key)	integer	Ідентифікатор сторінки мережі Twitter
Tweet	text	varchar(500)	Текст допису мережі Twitter
Tweet	published	date	Дата публікації допису мережі Twitter
Tweet	link	varchar(500)	Посилання на допис мережі Twitter
TweetWordCount	id (primary key)	integer	Ідентифікатор частоти токену в дописі мережі Twitter
TweetWordCount	tweet_id (foreign key)	integer	Ідентифікатор допису мережі Twitter
TweetWordCount	word_id (foreign key)	integer	Ідентифікатор токену
TweetWordCount	count	integer	Частота токену в дописі мережі Twitter

NewsWordCount	id (primary key)	integer	Ідентифікатор частоти токену в дописі мережі Twitter
NewsWordCount	article_id (foreign key)	integer	Ідентифікатор статті новинного сайту
NewsWordCount	word_id (foreign key)	integer	Ідентифікатор токену
NewsWordCount	count	integer	Частота токену в статті новинного сайту
Word	id (primary key)	integer	Ідентифікатор токену
Word	word	varchar(100)	Токен
ArtilceMetricsDynamics	id (primary key)	integer	Ідентифікатор метрик статті
ArtilceMetricsDynamics	article_id (foreign key)	integer	Ідентифікатор статті новинного сайту
ArtilceMetricsDynamics	data	JSON	Поле для даних в форматі ключ-значення
TweetMetricsDynamics	id (primary key)	integer	Ідентифікатор метрик допису
TweetMetricsDynamics	tweet_id (foreign key)	integer	Ідентифікатор допису мережі Twitter
TweetMetricsDynamics	data	JSON	Поле для даних в форматі ключ-значення

2.4 Аналіз безпеки даних

Дані про новини та дописи мережі Twitter збергіються в базу даних застосунку через API-інтерфейси й вже готові оболонки для роботи з ним. А також за допомогою веб-скрапінгу, що також реалізується з допомогою сторонніх бібліотек. Основою гарантії безпеки даних є з’єднання по надійних

мережевих протоколах.

2.5 Висновки по розділу

В даному розділі вже було детально спроектовано та описано архітектуру застосунка. Виокремлено всі сутності бази даних та встановлено між ними зв’язки. Бізнес-процеси були відображені у форматі діаграм BPMN.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Контроль та аналіз правильності функціонування розробленого програмного забезпечення – один з найважливіших етапів життєвого циклу розробки. Він визначає рівень відповідності між теоретично описаною та фактичною поведінкою програми.

Так як якість не можна вимірювати у чисельних показниках, вона є суб’єктивним поняттям. Етап тестування полягає у перевірці результатів кінцевого набору тестів, що визначається розробником чи специфікацією.

Будь-який план тестування складається з набору функцій, типів тестів та наборів програмних засобів, що зазнають тестування.

Даний план, що здійснює аналіз якості включає перевірку правильності та коректності роботи усіх складових продукту: від веб-скрапера до адаптивного інтерфейсу користувача.

Будуть перевірені результати наступних функцій:

- веб-скрапінг публікацій;
- отримання даних по API;
- збереження побудованих графіків;
- отримання агрегованих списків новин;
- отримання рейтингових списків джерел новин;
- отримання інформації за вказаний часовий проміжок;
- вибір джерела для здійснення аналітики.

Визначені наступні компоненти, що будуть протестовані:

- веб-скрапінг публікацій з новинних сайтів;
- веб-скрапінг інформації про новинний ресурс;
- отримання останніх твітів та метрики по них через Twitter API;

– отримання статистичних метрик обраної сторінки через Twitter API;

– отримання рейтингового списку новинних ресурсів за статистичними показниками;

– отримання агрегованого спику новин за актуальністю;

– отримання агрегованого списку новин за популярністю;

– отримання агрегованого списку новин за датою публікації;

– завантаження зображення побудованого графіку трендів новин;

– вибір часового діапазону по якому здійснюється агрегація новин.

3.2 Методики тестування

Будемо використовувати такі тестові методики:

– інтеграційна методика;

– модульна методика;

– навантажувальна методика.

3.2.1 Тестування за інтеграційною методикою

Цей метод тестування допоможе знайти помилки у інтеграції та взаємодії між логічними частинами застосунку:

– взаємодія серверної частини та та бази даних;

– взаємодія серверної та клієнтської частин;

– взаємодія серверної частини та модулів скрапінгу.

3.2.2 Тестування за модульною методикою

Цей метод тестування допоможе знайти помилки у роботі прикладного інтерфейсу серверної частини додатку:

– відсутність обробників для неописаних маршрутів;

– некоректна реакція на вкладені маршрути;

– відкритість для всіх внутрішніх методів.

3.2.3 Тестування за навантажувальною методикою

Цей метод тестування допоможе знайти помилки у роботі веб-додатку за умов значного мережевого та апратного навантаження:

– збереження з’єднання при ненормованих навантаженнях;
– виконання операції з базою даних.

3.3 Критерії проходження тестування

3.3.1 Тестування за інтеграційною та модульною методиками

Тестування інтеграційним та модульними методами буде пройденим у випадку, коли виконається кожен з пунктів відповідного тестового методу. Навіть один непройдений тест нівелює попередньо позитивний результат всього етапу тестування.

3.3.2 Тестування за навантажувальною методикою

Тестування навантажень буде пройденим у випадку, коли виконається кожен з пунктів тестового методу. Параметрами у ньому є кількість одночасних запитів від серверу до сторонніх API та швидкість обробки вхідних даних. Навіть один непройдений тест нівелює попередньо позитивний результат всього етапу тестування.

3.4 Процес тестування

3.4.1 Дані до тестів

Вхідні дані тесту інтеграції – множина можливих повідомлень, які отримуються системою від сторонніх API. Вихідні дані – результати роботи системи у відповідних ситуаціях.

Вхідні дані тесту компонентів – множина параметрів та відповідна їй множина визначених результатів, яка є вихідними даними цього етапу тестування.

Вхідні дані тесту швидкодії – множина даних, що імітує використання програмного забезпечення у продакшені. Вихідні дані – кількісні показники швидкості обробки, відсоткові дані по навантаженню, статистика.

3.4.2 Задачі тесту

Мета виконання кожного з тестів – перевірка роботи програми в умовах виконання, виявлення недоліків і помилок.

3.5 Вимоги до середовища

3.5.1 Апаратна частина

Апаратна частина середовища для виконання тестів має відповідати вимогам, що поставлені для запуску та розгортання програмного засобу.

3.5.2 Вимоги до безпеки

Для успішного проведення тестування на апаратній платформі повинна бути встановлена операційна система на базі Linux або macOS. Обов'язковими факторами є: встановлені інтерпритатор мови Python 3.5+, пакетний менеджер pip та СУБД PostgreSQL.

3.5.3 Інструменти

Для прискорення, спрощення та автоматизації процесу тестування та перевірки використаємо такі утиліти та програми:

- pytest;
- unittest;
- Jest.

3.6 Опис контрольного прикладу

В таблицях 3.1-3.4 детально опишемо контрольнф приклади тестування різних складових готового продукту.

Таблиця 3.1 – Обробка некоректних маршрутів

Мета тесту	Перевірка коректної обробки при заданні некоректного маршруту
Початковий стан	Відкриті програмні засоби
Вхідні дані	Некоректний маршрут
Схема проведення тесту	Перейти до пошукового рядку, та додати до домену сайту явно не коректний маршрут
Очікуваний результат	Застосунок повідомить про некоректно введений маршрут та запропонує схожі варіанти або ж повернутися на попередню сторінку

Таблиця 3.2 – Завантаження зображення графіку трендів

Мета тесту	Перевірка можливості завантаження зображення побудованого графіку трендів
Початковий стан	Відкриті програмні засоби
Вхідні дані	Вказаний часовий діапазон та джерело новин
Схема проведення тесту	Перейти до сторінки аналізу, обрати часовий діапазон та джерело новин для аналізу, натиснути кнопку «Графік трендів» Далі повинна з’явитися кнопка «Зберегти», потрібно натиснути її
Очікуваний результат	Після задання параметрів застосунок будує графік, з'являється кнопка для збереження, після її натиснення

Продовження таблиці 3.2

	зображення завантажується на персональний комп’ютер

Таблиця 3.3 – Перегляд сторінок сайту

Мета тесту	Перевірка швикодії застосунку
Початковий стан	Відкриті програмні засоби
Вхідні дані	Почерговий запит на відкриття різних сторінок
Схема проведення тесту	По черзі відвідати кожну сторінку програмного комплексу та протестувати наявний функціонал
Очікуваний результат	Застосунок відображає всі свої сторінки за мінімільний час очікування, що не перевищує декількох секунд

Таблиця 3.4 – Перевірка валідації даних

Мета тесту	Перевірка валідації даних при скрапінгу новинних сайтів
Початковий стан	Відкриті програмні засоби
Вхідні дані	Обрані дані на головній сторінці, наприклад дата та текст заголовка новини
Схема проведення тесту	Перейти за посиланням на джерела обраних новин та перевірити на коректність відображення обраних даних на сторінці застосунку, порівнняно з даними джерел

Продовження таблиці 3.4

Очікуваний результат	Обрані дані успішно перевірені, дата та час відображаються коректно, згідно відповідного часового поясу

3.7 Висновки по розділу

Під час тестування програмного забезпечення були викоремлені та детально описані методики тестування, критерії їх проходження та інструменти, що для цього використовувалися. Результатом тестування стала задовільна якість застосунку та його придатність до користування.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Повноцінне розгортання розробленого комплексу та коректна його робота з великими навантаженнями вимагає наступних кроків:

– встановлення бібліотек (написаних на Python) функціонал яких полягає в здійсненні веб-скрапінгу (BeautifulSoup, Scrapy);

– реєстрація та створення своєї власної Twitter App, що надає унікальні ключі доступу;

– встановлення стеку Python/Django та супутньої інфраструктури;

– встановлення СУБД PostgreSQL.

4.1.1 Встановлення стеку Python/Django

Спочатку встановимо Python версії 3-х .

Рисунок 4.1 – Команада терміналу для встановлення Python

Далі встановимо пакетний менеджер «pip».

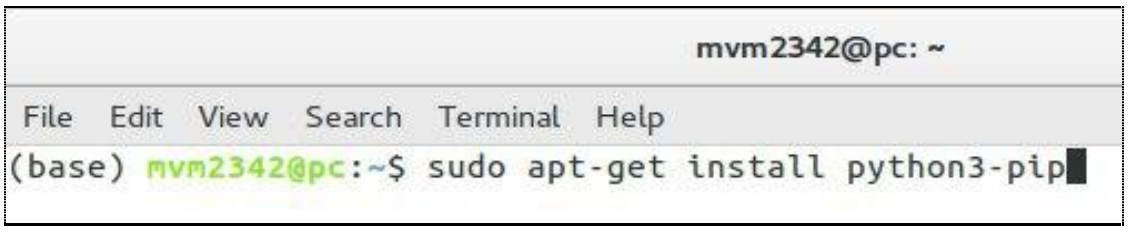

Рисунок 4.2 – Команада терміналу для встановлення «pip»

Наступним кроком встановимо весь необхідний стек Python/Django, для підтримки розгортання написаного продукту на даному тезнологічному наборі інструментів.

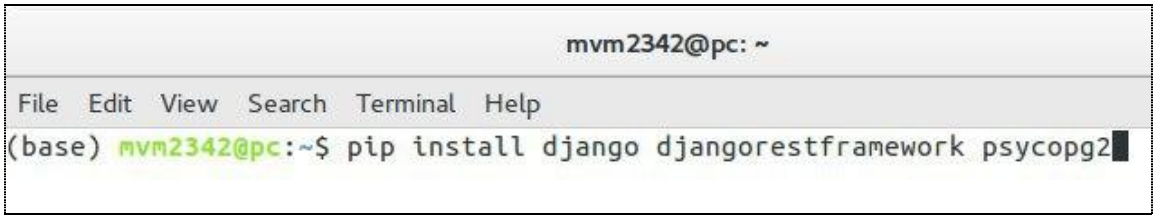

Рисунок 4.3 – Команади терміналу для встановлення стеку Python/Django

4.1.2 Встановлення бібліотек для веб-скрапінгу

Для встановлення будь-яких бібліотек, зокрема бібілотек для скрапінгу, написаних на мові програмування Python необхідний вже встановлений менеджер пакетів «pip».

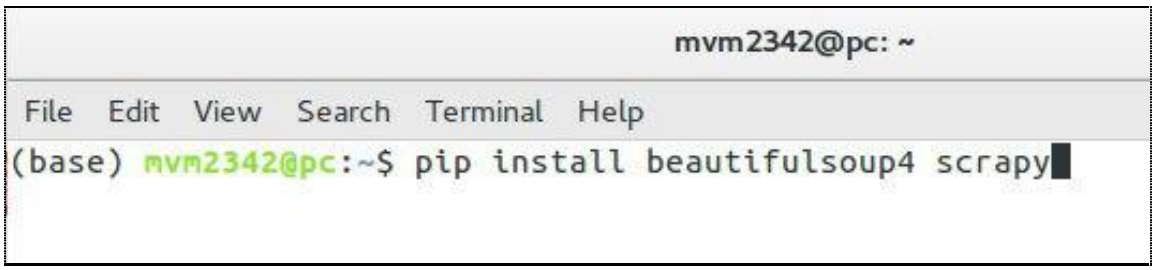

Рисунок 4.4 – Команада терміналу для встановлення бібліотек для скрапінгу

4.1.3 Реєстрація та створення Twitter App

Для початку роботи з Twitter API необхідно зареєструвати свою власну Twitter App.

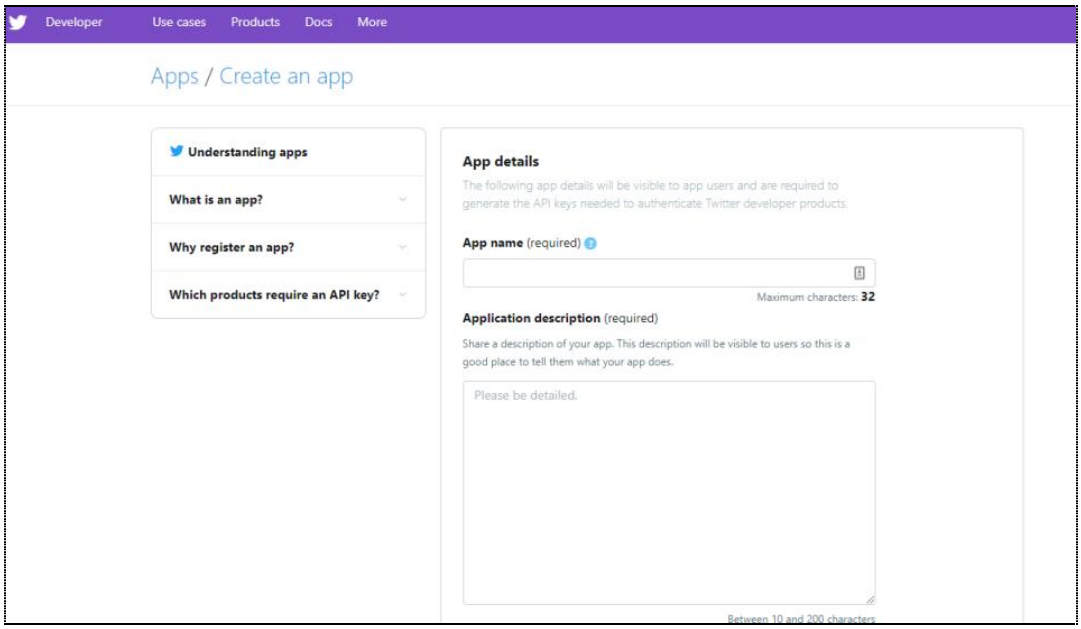

Рисунок 4.5 – Головна сторінка створення Twitter App

4.1.4 Встановлення СУБД PostgreSQL

В якості СУБД будемо використовувується PostgreSQL, нижче приклад встановлення.

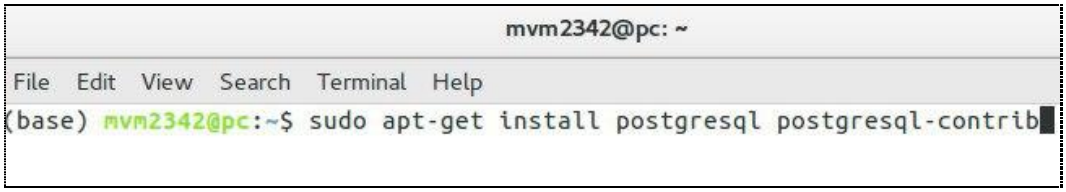

Рисунок 4.6 – Команада терміналу для встановлення PostgreSQL

4.1.5 Запуск сервера

Фінальним кроком у розгортанні програмного забезпечення буде запуск сервера, в даному випадку на локалці.

Рисунок 4.7 – Команада терміналу для запуску

4.2 Робота з програмним забезепеченням

Як використовувати програму звичайному користувачеві та який функціонал вона підтримує, вказано в інструкціях по роботі програмного забезпечення, яка наведена в керівництві користувача.

4.3 Висновки по розділу

В даному розділі були розписані всі необхідні кроки для встановлення залежностей програмного комплексу та його розгортання. Інструкції ж для звичайних користувачів наведені в керівництві користувача.

ВИСНОВКИ

В рамках даної дипломної роботи було проаналізовано предметну область, розроблено та втілено створену архітектуру програмного забезпечення. По завершенню розробки було реалізовано план тестування та відповідно до нього протестовано реалізований програмний засіб.

Також створена детальна інструкція з підготовки середовища для розгортання та запуску програмного засобу на апаратній платформі і створено інструкції з використання для програміста та адміністратора.

ПІд час роботи над даним дипломним проектом були поглиблені та застосовані знання з реляційних баз даних, розробки на мові програмування Python, архітектури програмного забезпечення, розробки інтерфейсів, були вивчені нові інструменти тестування.

Результатом виконання роботи було створення повноцінного робочого програмного додатку, що готовий до використання. Важливою можливістю є розширення набору використовуваних новинних ресурсів та їх адаптація для подальшого скрапингу та парсингу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Twitter [Електронний ресурс] – Режим доступу: https://twitter.com/
2. Документація модуля Psycopg [Електронний ресурс] – Режим доступу: http://initd.org/psycopg/docs/
3. Документація модуля Tweepy [Електронний ресурс] – Режим доступу: https://www.tweepy.org/
4. Документація мови Python [Електронний ресурс] – Режим доступу: https://docs.python.org/3/
5. Документація фреймворку Django [Електронний ресурс] – Режим доступу: https://www.djangoproject.com/
6. Web scraping [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Web_scraping
7. Linear trend estimation [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Linear_trend_estimation
8. How to use Scrapy with Django Application [Електронний ресурс] – Режим доступу: https://medium.com/@ali_oguzhan/how-to-use-scrapy-with-django-application/
9. Celery: лучшие практики [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/269347/
10. Документація модуля Beautiful Soup [Електронний ресурс] – Режим доступу: https://pypi.org/project/beautifulsoup4/

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

Комплекс задач агрегації та аналізу новин

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк, 1284 Кб

(Обсяг програми (документа) , арк.,) Кб)

models.py

```
import pytz
import requests
import datetime
import newspaper
from django.contrib.postgres.fields import JSONField
from django.db import models

class NewsSite(models.Model):
      username = models.CharField(max_length=100, null=True, blank=True)
      fullname = models.CharField(max_length=200, null=True, blank=True)
      rsslink = models.CharField(max_length=400, null=True, blank=True)
      link = models.CharField(max_length=200, null=True, blank=True)
      logo = models.CharField(max_length=400, null=True, blank=True)
      bio = models.CharField(max_length=1000, null=True, blank=True)

class TwitterProfile(models.Model):
      username = models.CharField(max_length=100, null=True, blank=True)
      fullname = models.CharField(max_length=200, null=True, blank=True)
      link = models.CharField(max_length=200, null=True, blank=True)
      logo = models.CharField(max_length=400, null=True, blank=True)
      bio = models.CharField(max_length=1000, null=True, blank=True)
      subscribers = models.IntegerField(null=True, blank=True)

class Tweet(models.Model):
      text = models.CharField(max_length=500, null=True, blank=True)
      retweets = models.IntegerField(null=True, blank=True)
      comments = models.IntegerField(null=True, blank=True)
      likes = models.IntegerField(null=True, blank=True)
      published = models.DateTimeField(auto_now_add=True, blank=True)
      link = models.CharField(max_length=400, null=True, blank=True)
      twitterprofile = models.ForeignKey('TwitterProfile',
on_delete=models.CASCADE)
class Article(models.Model):
```

```
    text = models.CharField(max_length=10000, null=True, blank=True)
    title = models.CharField(max_length=500, null=True, blank=True)
    link = models.CharField(max_length=500, null=True, blank=True)
    comments = models.IntegerField(null=True, blank=True)
    views = models.IntegerField(null=True, blank=True)
    published = models.DateTimeField(auto_now_add=True, blank=True)
    newssite = models.ForeignKey('NewsSite', on_delete=models.CASCADE)

class ArticleWordCount(models.Model):
    article = models.ForeignKey('Article', on_delete=models.CASCADE)
    word = models.ForeignKey('Word', on_delete=models.CASCADE)
    count = models.IntegerField(null=True, blank=True)

class TweetWordCount(models.Model):
    tweet = models.ForeignKey('Tweet', on_delete=models.CASCADE)
    word = models.ForeignKey('Word', on_delete=models.CASCADE)
    count = models.IntegerField(null=True, blank=True)

class Word(models.Model):
    word = models.CharField(max_length=200, null=True, blank=True)

class ArticleMetricsDynamics(models.Model):
    article = models.ForeignKey('Article', on_delete=models.CASCADE)
    data = JSONField(null=True, blank=True)

class TweetMetricsDynamics(models.Model):
    tweet = models.ForeignKey('Tweet', on_delete=models.CASCADE)
    data = JSONField(null=True, blank=True)
```

parsers.py

```
import pytz
import requests
import datetime
import newspaper
from bs4 import BeautifulSoup
from news.models import *

class DBHelper(object):
      @staticmethod
      def get_saved_sites():
            try:
                  sites = NewsSite.objects.all()
            except:
                  sites = []
            return sites
      @staticmethod
      def get_saved_articles():
            try:
                  articles = Article.objects.all()
            except:
                  articles = []
            return articles
      @staticmethod
      def get_saved_articles_links():
            articles = DBHelper.get_saved_articles()
            articles_links = [article.link for article in articles]
            return articles_links
      @staticmethod
      def get_saved_sites_links():
            sites = DBHelper.get_saved_sites()
            sites_links = [site.link for site in sites]
            return sites_links
```

```
      @staticmethod
      def get_sites_rss_links():
            sites = DBHelper.get_saved_sites()
            sites_rss_links = [site.rss_link for site in sites]
            return sites_rss_links


class DBSaver(object):
      @staticmethod
      def save_articles(articles):
            for article in articles:
                  instance = Article(
                        title=article['title'],
                        published=article['published'],
                        link=article['link'],
                        text=article['text'],
                        newssite=article['site'],
                        views=article['views']
                  )
                  instance.save()
      @staticmethod
      def engine():
            obj_up = ParserUP()
            articles = obj_up.get_articles()
            DBSaver.save_articles(articles)
            obj_up = ParserCN()
            articles = obj_up.get_articles()
            DBSaver.save_articles(articles)
            obj_up = ParserTSN()
            articles = obj_up.get_articles()
            DBSaver.save_articles(articles)
      @staticmethod
      def update_article(data):
      Article.objects.filter(link=data['link']).update(views=data['views'
])
```

```
            print('UPDATe')
            print(data['views'])

class ParserUP(object):
      def __init__(self, name='PJPц'):
            self.name = name
            self.site = NewsSite.objects.filter(username=self.name)[0]
            self.start_url = self.site.rsslink

      def get_news_block(self, soup):
            block_news_all = soup.find('div', attrs={'class': 'news
news_all'})
            block_articles = block_news_all.findAll('div', attrs={'class':
'article'})
            return block_articles

      def get_news_links(self):
            page = requests.get(self.start_url)
            soup = BeautifulSoup(page.text, 'html.parser')
            articles = self.get_news_block(soup)
            links_list = []
            for article in articles:
                  link_tag = article.find('a')
                  link_href = link_tag['href']
                  if link_href.startswith('/news/'):
                        link_href = 'https://www.pravda.com.ua' + link_href
                        links_list.append(link_href)
            return links_list

      def get_article_data(self, link):
            article = newspaper.Article(link)
            article.download()
            article.parse()
```

```
            article_data = {
                  'title': article.title,
                  'text': article.text,
                  'date': article.publish_date,
            }
            return article_data

      def get_article_stat(self, link):
            page = requests.get(link)
            soup = BeautifulSoup(page.text, 'html.parser')
            source = soup.find('div', attrs={'class': 'post__views'})
            text = source.text
            count = [int(s) for s in text.split() if s.isdigit()][-1]
            source = soup.find('div', attrs={'class': 'post_news__date'})
            text = source.text
            time = text.split(', ')[-1]
            time_hour, time_minute = time.split(':')
            article_stat = {
                  'views': count,
                  'time_hour': int(time_hour),
                  'time_minute': int(time_minute),
            }
            return article_stat

      def get_articles(self):
            links = self.get_news_links()
            saved_links = DBHelper.get_saved_articles_links()
            articles = []
            for link in links:
                  if link not in saved_links:
                        stat = self.get_article_stat(link)
                        data = self.get_article_data(link)
                        published = datetime.datetime(
                              data['date'].year,
```

```
                        data['date'].month,
                        data['date'].day,
                        stat['time_hour'],
                        stat['time_minute']
                    )
                    articles.append({
                        'title': data['title'],
                        'text': data['text'],
                        'views': stat['views'],
                        'published': published,
                        'link': link,
                        'site': self.site,
                    })
                else:
                    data = self.get_article_stat(link)
                    data['link'] = link
                    data['site'] = self.site
                    DBSaver.update_article(data)
            return articles

class ParserCN(object):
    def __init__(self, name='P¦Pќ'):
        self.name = name
        self.site = NewsSite.objects.filter(username=self.name)[0]
        self.start_url = self.site.rsslink

    def get_news_links(self):
        page = requests.get(self.start_url)
        soup = BeautifulSoup(page.text, 'html.parser')
        articles = soup.findAll('article', attrs={'class': 'item'})
        links_list = []
        for article in articles:
            try:
```

```
                    links = article.find('div', attrs={'class':
'anounce'})
                    link = links.find('a')
                    links_list.append(link['href'])
               except:
                    pass
          return links_list

     def get_article_stat(self, link):
          page = requests.get(link)
          soup = BeautifulSoup(page.text, 'html.parser')
          source = soup.find('span', attrs={'class': 'info'})
          views = int(source.text)
          source = soup.find('time', attrs={'class': 'published
dateline'})
          text = source.text
          published = datetime.datetime.strptime(text, '%d.%m.%y %H:%M')

          article_stat = {
               'views': views,
               'published': published
          }
          return article_stat

     def get_article_data(self, link):
          article = newspaper.Article(link)
          article.download()
          article.parse()
          article_data = {
               'title': article.title,
               'text': article.text,
          }
          return article_data
```

```
        def get_articles(self):
            links = self.get_news_links()
            saved_links = DBHelper.get_saved_articles_links()
            articles = []
            for link in links:
                if link not in saved_links:
                    data = self.get_article_data(link)
                    stat = self.get_article_stat(link)
                    articles.append({
                        'title': data['title'],
                        'text': data['text'],
                        'views': stat['views'],
                        'published': stat['published'],
                        'link': link,
                        'site': self.site,
                    })
                else:
                    data = self.get_article_stat(link)
                    data['link'] = link
                    data['site'] = self.site
                    DBSaver.update_article(data)
            return articles

class ParserTSN(object):

        def __init__(self, name='РўРЎРќ'):
            self.name = name
            self.site = NewsSite.objects.filter(username=self.name)[0]
            self.start_url = self.site.rsslink

        def get_news_links(self):
            page = requests.get(self.start_url)
            soup = BeautifulSoup(page.text, 'html.parser')
```

```
            divs = soup.findAll('div', attrs={'class': ['o-jumbotron-
primary', 'js-theme-box']})
            articles = []
            for div in divs:
                  articles += div.findAll('article')
            i = 0
            links_list = []
            for article in articles:
                  links = article.findAll('a', attrs={'class':'u-url u-
uid'})
                  for a in links:
                        links_list.append(a['href'])
                        i += 1
                        print(i)
            return links_list

      def get_article_data(self, link):
            article = newspaper.Article(link)
            article.download()
            article.parse()
            article_data = {
                  'title': article.title,
                  'text': article.text
            }
            return article_data

      def get_articles(self):
            links = self.get_news_links()
            saved_links = DBHelper.get_saved_articles_links()
            articles = []
            for link in links:
                  if link not in saved_links:
                        data = self.get_article_data(link)
                        stat = self.get_article_stat(link)
```

```
                    if stat is not None:
                        articles.append({
                            'title': data['title'],
                            'text': data['text'],
                            'published': published,
                            'views': stat['views'],
                            'link': link,
                            'site': self.site,
                        })
                else:
                    data = self.get_article_stat(link)
                    data['link'] = link
                    data['site'] = self.site
                    DBSaver.update_article(data)
        return articles
```

handlers.py

```
class IndexViewHandler(object):

    def __init__(self, arg=None):
        self.arg = arg

    def get_most_viewed_news(self):
        articles = Article.objects.all().order_by('-views')
        for article in articles:
            str_pub = article.published
            str_pub = str_pub.strftime("%H:%M")
            article.published = str_pub.split(' ')[-1]
        return articles[0:10]

    def get_actual_news_head(self):
        articles = Article.objects.order_by('-published', '-views')
        return articles[0:12]

    def get_latest_news(self):
```

```
            articles = Article.objects.all().order_by('-published')
            for article in articles:
                  str_pub = article.published
                  article.published = str_pub.split(' ')[-1]
            return articles[0:10]


class AboutViewHandler(object):

      def __init__(self, arg=None):
            self.arg = arg

      def get_sites_list(self):
            sites = NewsSite.objects.all()
            return sites

      def get_sites_stat(self):
            stat = list()
            for site in self.get_sites_list():
                  articles = Article.objects.filter(site_id=site.id)

                  views = 0
                  for article in articles:
                        views += article.views

                  stat.append({'id': site.id, 'views': views})

            return stat
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ АГРЕГАЦІЇ ТА АНАЛІЗУ НОВИН

Технічне завдання

КПІ.ІП-5211.045440.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.М. Молявчик

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Комплекс задач агрегації та аналізу новин

Галузь застосування: Наведене технічне завдання поширюється на розробку програмного забезпечення «Комплекс задач агрегації та аналізу новин» КПІ.ІП-5211.045440, котра використовується для агрегації новинної повістки новинних сайтів та сторінок соціальної мережі Twitter, а також їх статистичного аналізу та призначена для звичайних користувачів, що зацікавлені в швидкому отриманні актуальних новин з різних джерел.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки Комплексу задач агрегації та аналізу новин є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання користувачами Інтернету, що прагнуть отримувати актуальні новини, як з новинних сайтів так і з соціальної мережі Twitter.

Метою розробки є вирішення проблеми відсутності успішних працюючих аналогів даної розробки, яка полягає в готовому продукті агрегації та аналізу новинної повістки.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- перегляд сторінок, що містять агреговані списки новин;
- статистичний аналіз ресурсів за обраний часовий діапазон;
- дослідження трендів ресурсів за обраний часовий діапазон;
- збереження зображень побудованих графіків трендів.

4.1.2 Розробку виконати на платформі Linux.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

Даний продукт не потребує регулярного обслуговування, адже всі бізнес-процеси в застосунку здійснюються автоматично, а втручання адміністратора необхідне тільки за надзвичайних умов.

4.3.3 Обслуговуючий персонал

Для обслуговування достатньо одного адміністратора, для можливого відновлення системи або ж виправлення некоректної її роботи.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

Тип процесору x86_64.

Об‘єм ОЗП 1024 Мб.

Достатній об’єм жорсткого диску 1 Гб.

Клавіатура.

ЖК-дісплей.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейств Unix.

4.5.2 Вхідні дані є набором параметрів переданих в результаті взаємодії користувача та графічного інтерфейсу.

4.5.3 Результати повинні бути представлені в наступному форматі: таблиці з текстовими даними при статистичному аналізі та зображення в форматі .png при дослідженні трендів новин.

4.5.4 Програмне забезпечення повинно надавати графічний інтерфейс користувача та REST API.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Спеціальні вимоги не пред’являються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування.

5.4 Графічна частина повинна бути виконана на 3 листах формату А3, котрі включаються до розділу «Графічні матеріали»:

5.4.1 Схема бази даних.

5.4.2 Схема структурна класів програмного забезпечення.

5.4.3 Схема структурна варіантів використання.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	18.03.2019	
2.	Аналіз існуючих методів розв’язання задачі	25.03.2019	
3.	Постановка та формалізація задачі	25.03.2019	Технічне завдання
4.	Аналіз вимог до програмного забезпечення	01.04.2019	Вимоги до ПЗ, СС варіантів використання
5.	Алгоритмізація задачі	01.04.2019	
6.	Моделювання програмного забезпечення	08.04.2019	
7.	Обґрунтування використовуваних технічних засобів	15.04.2019	
8.	Розробка архітектури програмного забезпечення	22.04.2019	СС програмного забезпечення
9.	Розробка програмного забезпечення	29.04.2019	Тексти програмного забезпечення
10.	Налагодження програми	06.05.2019	Програма Тестування
11.	Виконання графічних документів	13.05.2019	Графічні матеріали
12.	Оформлення пояснювальної записки	20.05.2019	Пояснювальна записка
13.	Подання ДП на попередній захист	28.05.2019	
14.	Подання ДП рецензенту	03.05.2019	
15.	Подання ДП на основний захист	08.06.2019	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ АГРЕГАЦІЇ ТА АНАЛІЗУ НОВИН

Програма та методика тестування

КПІ.ІП-5211.045440.04.51

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____К.І. Ліщук

Виконавець:

_____В.М. Молявчик

Київ – 2019 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом тестових випробувань є програмний комплекс, що виконує задачі агрегації та аналізу новин, розроблений у вигляді веб-застосунку за допомогою стеку технологій Python/Django з використанням PostgreSQL, як СУБД, головним механізмом якого є веб-скрапінг за допомогою Scrapy Framework та Beautiful Soup 4.

2 МЕТА ТЕСТУВАННЯ

Процес тестування полягав у перевірці програмних засобів за наступними критеріями:

- функціонально коректна робота кожної сторінки веб-сервісу;
- зручність роботи з застосунком;
- коректність завантаження зображення побудованого графіку;
- швидкий та коректний веб-скрапінг новинних ресурсів;
- коректність відображення графічних візуалізацій;
- відповідність програмного комплексу до технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування було проведено з використанням технології Gray Box Testing, при якому внутрішнє облаштування програми нам відоме лиш частково. Припускається доступ до внутрішньої структури та алгоритмів роботи програмного забезпечення для написання максимально ефективних тест-кейсів, але саме тестування проводиться за допомогою техніки чорного ящика, тобто з позиції користувача.

Використовуються наступні методи тестування:

- модульна методика тестування;
- інтеграційна методика тестування;
- навантажувальна методика тестування.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується наступним набором інструментів:

– Jest;

– unittest;

– pytest.

Тестування працездатності, або ж надійності веб-застосунку перевіряється методами:

– тестування графічного інтерфейсу;

– тестування стабільності роботи при різних умовах;

– тестування сирцевого коду за допомогою Unit-тестів;

– динамічного ручного тестування у відповідності до функціональних вимог;

– тестування коректної роботи при над нормованих навантаженнях об’ємів інформації.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПЛАТФОРМА ДЛЯ СПІЛЬНОЇ РОБОТИ ВИКЛАДАЧА ТА СТУДЕНТА ПІД ЧАС ІСПИТУ

Керівництво користувача

КПІ.ІП-5226.045440.03.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____О.Д. Шателюк

Київ – 2019 року

Коли користувач вводить url-адресу даного веб-додатку у адресному рядку веб-браузера, він бачить головну сторінку з навігаційною панеллю, що зображена на рисунку 1.1.

Рисунок 1.1 – Панель навігації

Під навігаційного панеллю користувач бачить головну сторінку додатку, що включає у себе головний контент. Сторінка поділена на блоки, кожен з яких сегментує новини за певними категоріями. Для того, щоб класифікувати новину за певною категорію використовується декілька алгоритмів з аналізу тексту. Головна сторінка зображена на рисунку 1.2.

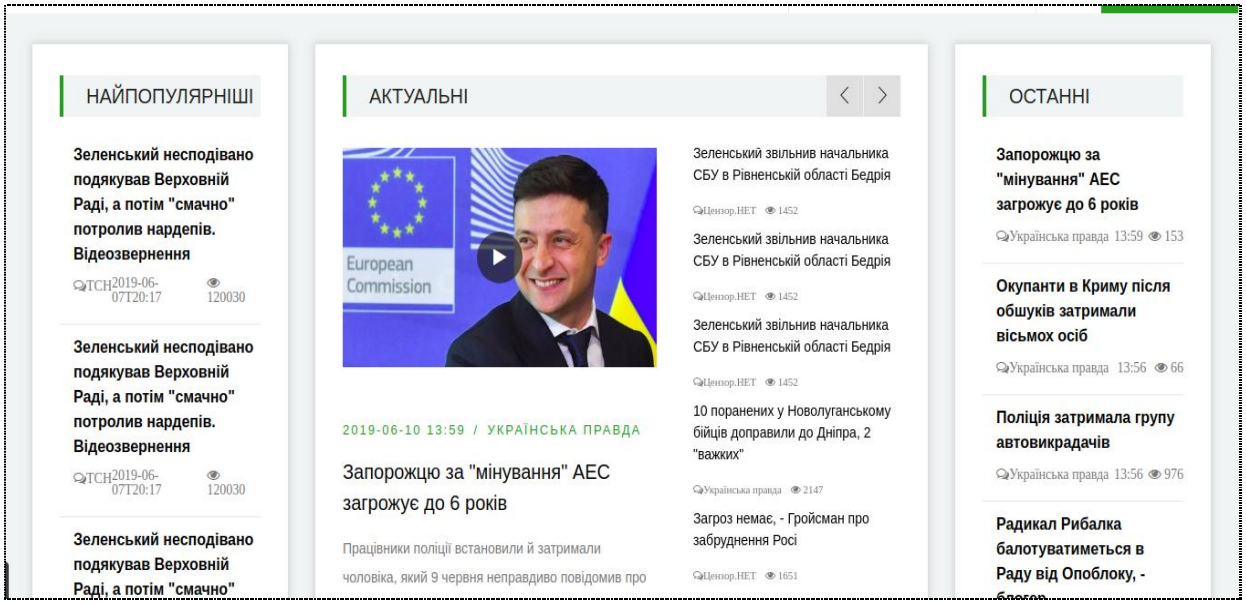

Рисунок 1.2 – Головна сторінка

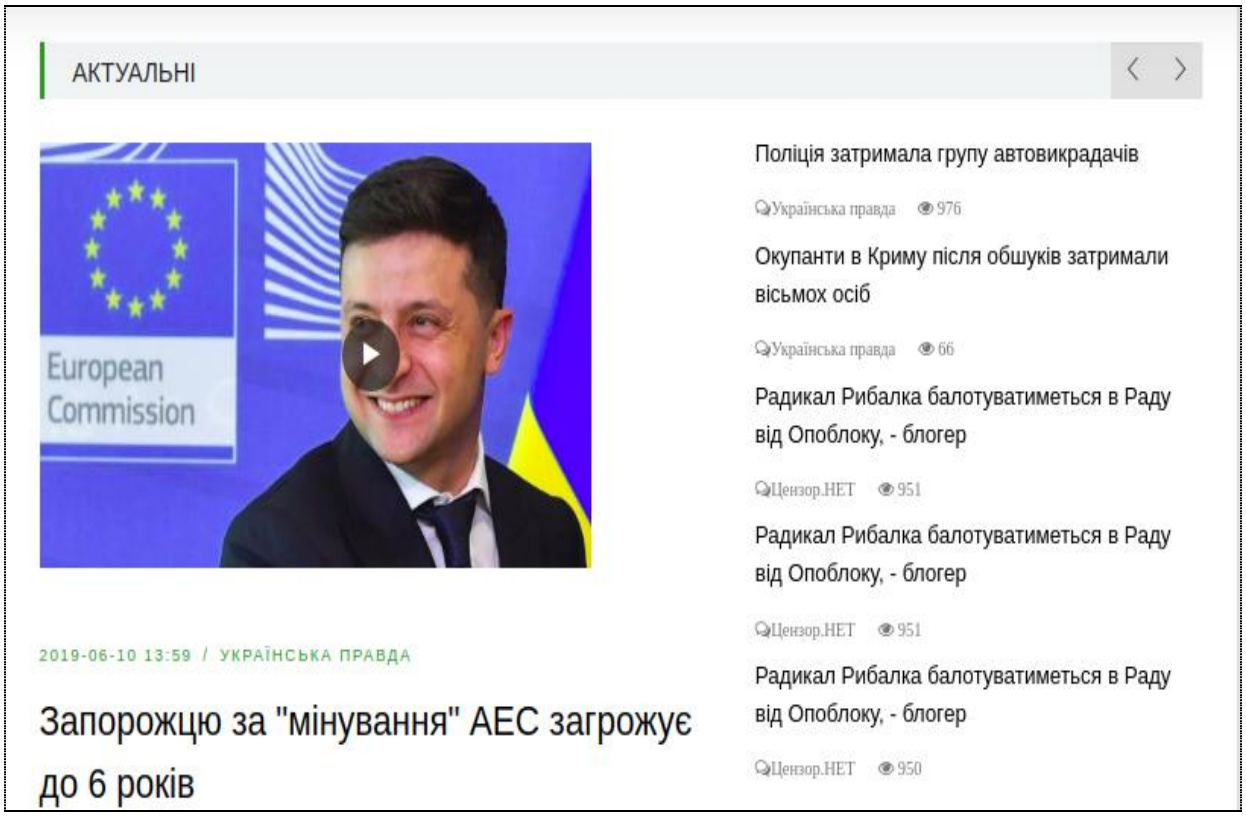

Рисунок 1.3 – Блок актуальних новин

У блоці актуальних новин користувач може переглядати список усіх новин, що є найбільш актуальними на даний час. Для визначення новин такого роду використовується алгоритм, що враховує динаміку росту обговорення, вподобань у соціальних мережах та кількість статей у різних медіа на одну і ту ж тему.

Новини перевіряються на унікальність для того щоб однакові новини з відмінним текстом чи заголовком не потрапляли до застосунку.

Рисунок 1.4 – Блок популярних новин

У блоці популярних новин користувач може переглядати список усіх новин, що створили у новинних медіа та соціальних мережах найбільший градус інтересу.

Новини перевіряються на унікальність для того щоб однакові новини з відмінним текстом чи заголовком не потрапляли до застосунку.

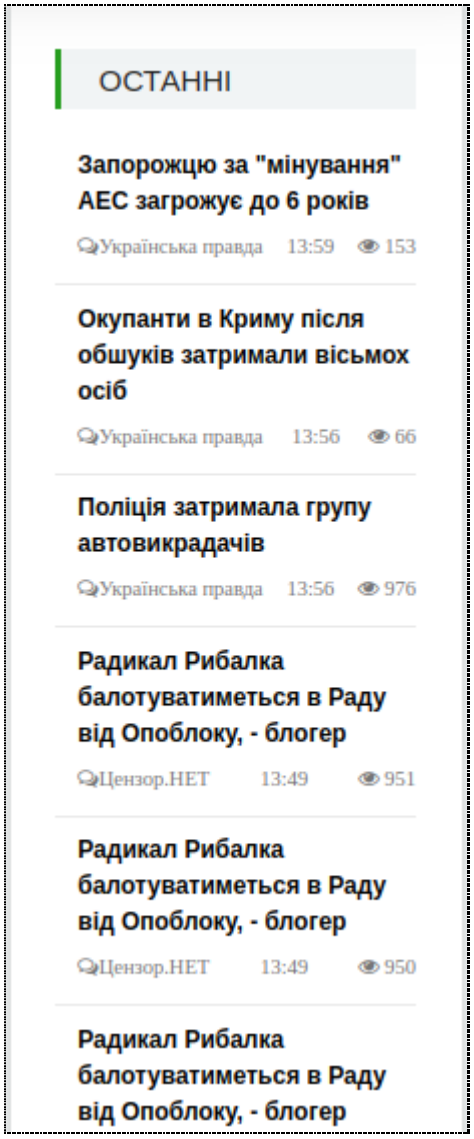

Рисунок 1.5 – Блок останніх новин

У блоці останніх новин користувач може переглядати список усіх новин, що з’явилися у новинних медіа та соціальних мережах впродовж недавнього часу.

Новини перевіряються на унікальність для того щоб однакові новини з відмінним текстом чи заголовком не потрапляли до застосунку.

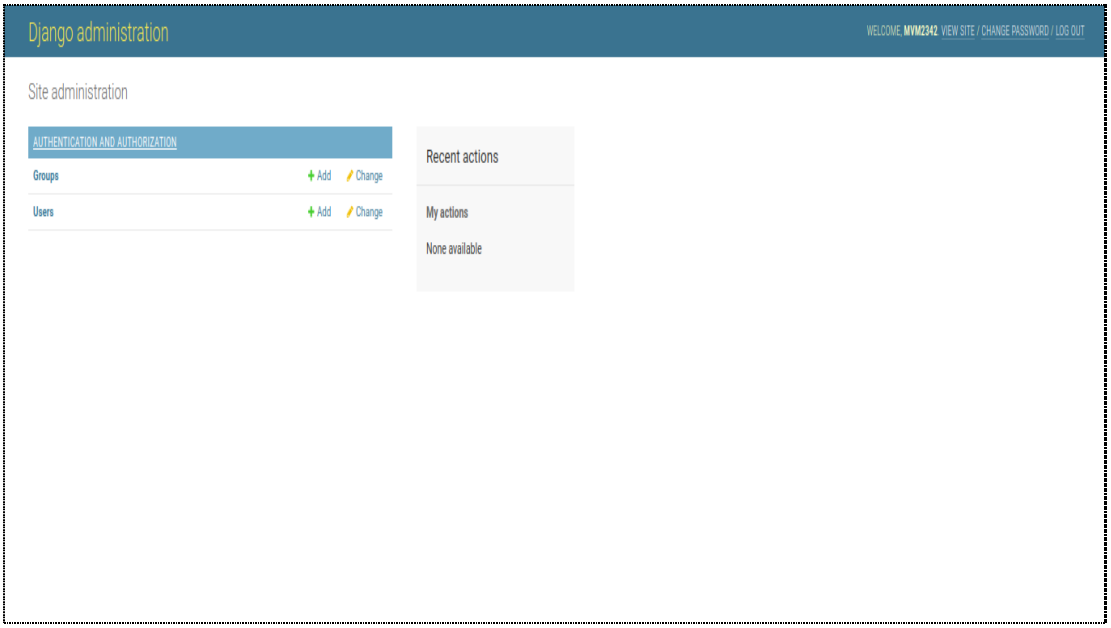

Рисунок 1.6 – Панель адміністратора

Перейшовши на сторінку з маршрутом «/admin» користувач потрапляє на панель адміністратора додатку. За наявності прав адміністратора користувач може модифікувати додаток та впливати на його роботу: додавати, видаляти, редагувати існуючі дані, а також додавати чи видаляти джерела новин.